IN THE BEGINNING

DEVELOPMENT AND OPERATIONS WERE FUNCTIONAL SILOS

Let there be

DevOps

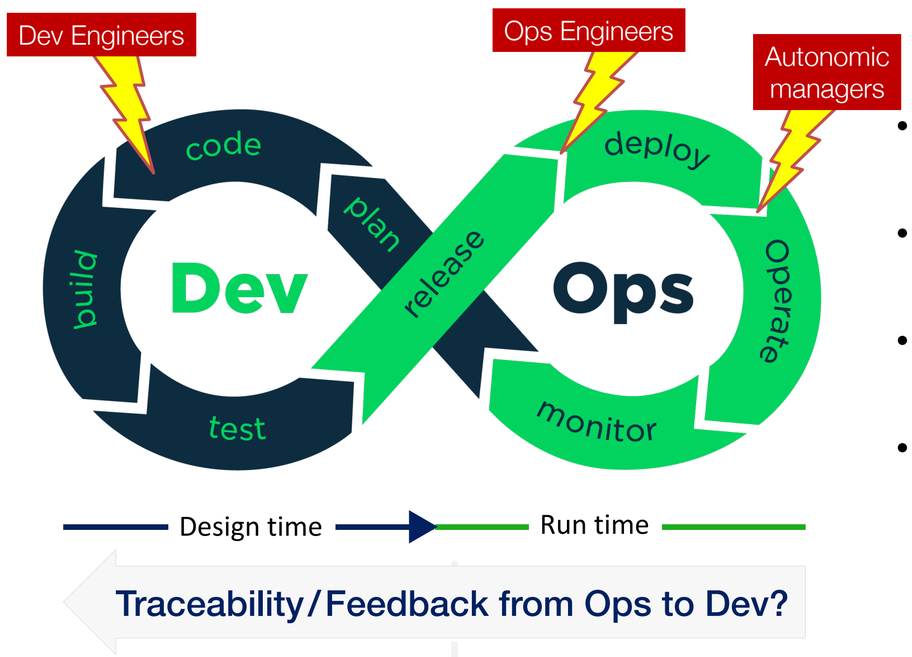
Integration · Measurement · Culture · Quality · Automation · Communication · Tools · Delivery · Improvement

The Motivation

Continuous Software Engineering

State of the Practice

- Developers and IT operators work on different environments/artefacts
- Continuous feedback lacks automation to impact software design
- Industry is moving towards eliminating the Ops role, while assigning more responsibilities to developers
- App code != deployment code. The latter is tested by deploying the app



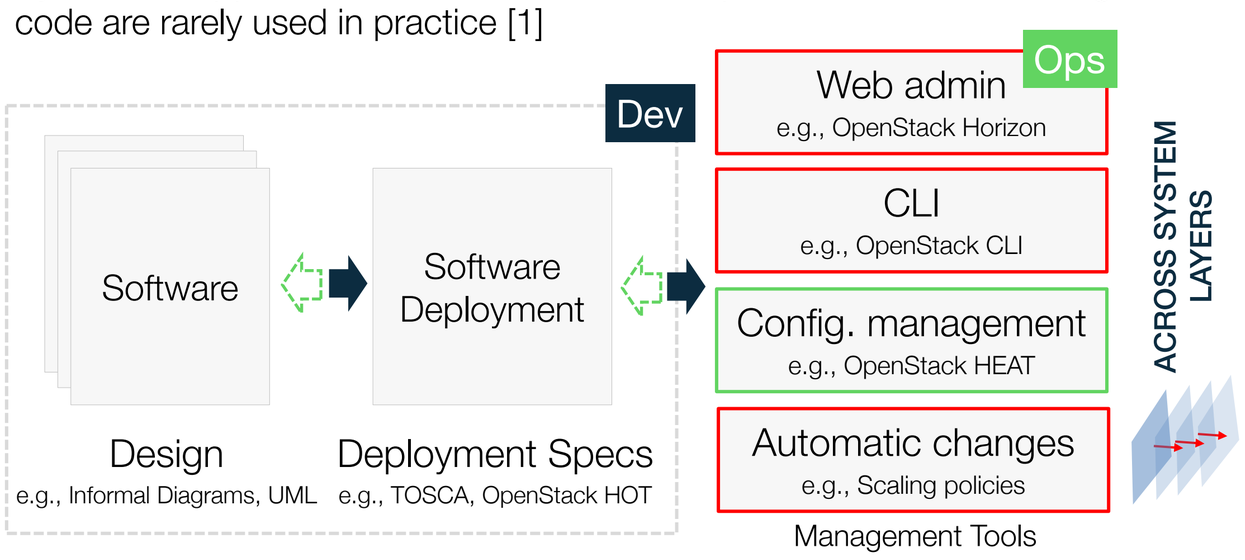
What's missing?

- How can we streamline continuous **feedback** to support the continuous development cycle?
- How can we enable autonomic capabilities to provide feedback from run time to design time?
- What kind of automation is needed to facilitate continuous **experimentation** and architecting?
- How can we assure **quality** for deployment code beyond static analysis, w/o deploying the system?

The Problem

Bidirectional Traceability

Systematic approaches to maintain the correspondence between design and code are rarely used in practice [1]



Continuous Integration (CI)

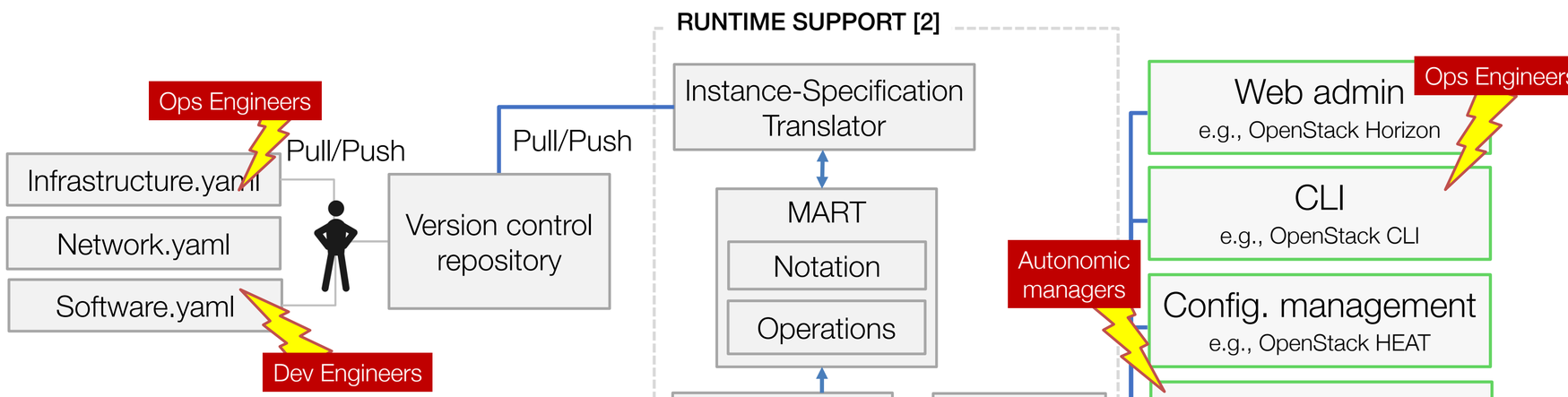
- Where are all these changes **logged**?
- How can they be **traced** back to their source?
- How and when are stakeholders **notified** about them?

CI is the answer, **isn't it?**

The Solution

Round-Trip Engineering — One Platform To Rule Them All

The infrastructure becomes a proxy to commit runtime actions



Benefits

- Specifications are always up to date
- Runtime data is fed back to development
- This framework enables further sync:
  - Design ↔ Deployment specs ↔ Running system
  - Runtime metrics → continuous experimentation specs

Considerations

CONTRIBUTION MODEL — Pragmatic Approach

Committer vs Contributor

- No update delay
- Less merge conflicts
- Unsupervised changes
- Update delay
- Time reviewing changes
- Frequent merge conflicts

CONFLICT RESOLUTION — Pragmatic Approach

Reliable vs Best effort

- Less prioritized changes are discarded
- Try to merge non-conflicting changes

CI PRINCIPLES

QA, build, deployment

- Automate the build?
- Make the build self-testing?
- **QA is open challenge**
- Automate deployment?

IBM Advanced Studies — Academic Applied Research Outcomes
University of Victoria
Miguel Jiménez, Hausi Müller — Victoria, Canada — {miguel, hausi}@uvic.ca
Gabriel Tamura — Cali, Colombia — gtamura@icesi.edu.co
UNIVERSIDAD ICESI

REFERENCES

1. Nugroho, Ariadi, and Michel RV Chaudron. "A survey of the practice of design--code correspondence amongst professional software engineers.". First International Symposium on Empirical Software Engineering and Measurement. IEEE, 2007.
2. Castañeda, Lorena. "Runtime Modelling for Smart User-centric Cyber-Physical-Human Applications". PhD thesis, 2017